

Appendix A

ADDLW **ADD literal to W**Syntax: `[label]: ADDLW k`Operands: $0 \leq k \leq 255$ Operation: $(W) + k \rightarrow W$

Status Affected: N,OV, C, DC, Z

Encoding:

0000	1111	kkkk	kkkk
------	------	------	------

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ADDLW 0x15`

Before Instruction

W = 0x10

After Instruction

W = 0x25

ADDWF **ADD W to f**Syntax: `[label] ADDWF f,d,a`Operands: $0 \leq f \leq 255$ d $\in [0,1]$ a $\in [0,1]$ Operation: $(W) + (f) \rightarrow \text{dest}$

Status Affected: N,OV, C, DC, Z

Encoding:

0010	01da	ffff	ffff
------	------	------	------

Description: Add W to register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWF REG, 0, 0`

Before Instruction

W = 0x17

REG = 0xC2

After Instruction

W = 0xD9

REG = 0xC2

ADDWFC **ADD W and Carry bit to f**Syntax: **[label] ADDWFC f,d,a**Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$ Operation: $(W) + (f) + (C) \rightarrow \text{dest}$

Status Affected: N,OV, C, DC, Z

Encoding:

0010	00da	ffff	ffff
------	------	------	------

Description:

Add W, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: **ADDWFC REG, 0, 1**

Before Instruction

Carry bit = 1
REG = 0x02
W = 0x4D

After Instruction

Carry bit = 0
REG = 0x02
W = 0x50

ANDLW **AND literal with W**Syntax: **[label] ANDLW k**Operands: $0 \leq k \leq 255$ Operation: $(W) \text{ AND } k \rightarrow W$

Status Affected: N,Z

Encoding:

0000	1011	kkkk	kkkk
------	------	------	------

Description:

The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: **ANDLW 0x5F**

Before Instruction

W = 0xA3

After Instruction

W = 0x03

ANDWF AND W with f

Syntax: [label] ANDWF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: (W) .AND. (f) \rightarrow dest

Status Affected: N,Z

Encoding:

0001	01da	ffff	ffff
------	------	------	------

Description: The contents of W are AND'ed with register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: ANDWF REG, 0, 0

Before Instruction

W = 0x17
REG = 0xC2

After Instruction

W = 0x02
REG = 0xC2

BC Branch if Carry

Syntax: [label] BC n

Operands: $-128 \leq n \leq 127$

Operation: if carry bit is '1'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0010	nnnn	nnnn
------	------	------	------

Description: If the Carry bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BC 5

Before Instruction

PC = address (HERE)

After Instruction

If Carry = 1;
PC = address (HERE+12)
If Carry = 0;
PC = address (HERE+2)

BCF **Bit Clear f**

Syntax: [label] BCF f,b,a

Operands: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$ Operation: $0 \rightarrow f \leftarrow b$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff
------	------	------	------

Description: Bit 'b' in register 'f' is cleared. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: BCF FLAG_REG, 7, 0

Before Instruction

FLAG_REG = 0xC7

After Instruction

FLAG_REG = 0x47

BN **Branch if Negative**

Syntax: [label] BN n

Operands: $-128 \leq n \leq 127$ Operation: if negative bit is '1'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn
------	------	------	------

Description: If the Negative bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BN Jump

Before Instruction

PC = address (HERE)

After Instruction

If Negative = 1;
PC = address (Jump)If Negative = 0;
PC = address (HERE+2)

BNC Branch if Not Carry

Syntax: [label] BNC n
Operands: $-128 \leq n \leq 127$
Operation: if carry bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0011	nnnn	nnnn
------	------	------	------

Description: If the Carry bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNC Jump

Before Instruction

PC = address (HERE)

After Instruction

If Carry = 0;
PC = address (Jump)
If Carry = 1;
PC = address (HERE+2)

BNN Branch if Not Negativ

Syntax: [label] BNN n
Operands: $-128 \leq n \leq 127$
Operation: if negative bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0111	nnnn	nnnn
------	------	------	------

Description: If the Negative bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNN Jump

Before Instruction

PC = address (HERE)

After Instruction

If Negative = 0;
PC = address (Jump)
If Negative = 1;
PC = address (HERE+2)

INV Branch if Not Overflow

Syntax: [label] BNV n

Operands: $-128 \leq n \leq 127$

Operation: if overflow bit is '0'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0101	nnnn	nnnn
------	------	------	------

Description: If the Overflow bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decod	Read literal 'n'	Process Data	No operation

Example: HERE BNV Jump

Before Instruction

PC = address (HERE)

After Instruction

If Overflow = 0;

PC = address (Jump)

If Overflow = 1;

PC = address (HERE+2)

BNZ Branch if Not Zero

Syntax: [label] BNZ n

Operands: $-128 \leq n \leq 127$

Operation: if zero bit is '0'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0001	nnnn	nnnn
------	------	------	------

Description: If the Zero bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BNZ Jump

Before Instruction

PC = address (HERE)

After Instruction

If Zero = 0;

PC = address (Jump)

If Zero = 1;

PC = address (HERE+2)

BRA **Unconditional Branch**

Syntax: `[label] BRA n`

Operands: $-1024 \leq n \leq 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1101	0nnn	nnnn	nnnn
------	------	------	------

Description: Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE BRA Jump`

Before Instruction

PC = address (HERE)

After Instruction

PC = address (Jump)

BSF **Bit Set f**

Syntax: `[label] BSF f,b,a`

Operands: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$

Operation: $1 \rightarrow f$

Status Affected: None

Encoding:

1000	bbba	ffff	ffff
------	------	------	------

Description: Bit 'b' in register 'f' is set. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BSF FLAG_REG, 7, 1`

Before Instruction

FLAG_REG= 0x0A

After Instruction

FLAG_REG= 0x8A

BTFSF Bit Test File, Skip if Clear

Syntax: [label] BTFSF f,b,a

Operands: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$ Operation: skip if $(f < b) = 0$

Status Affected: None

Encoding:

1011	bbba	ffff	ffff
------	------	------	------

Description:

If bit 'b' in register 'r' is 0 then the next instruction is skipped.

If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'r'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFSF FLAG, 1, 0
 FALSE :
 TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;
 PC = address (TRUE)
 If FLAG<1> = 1;
 PC = address (FALSE)

BTFSF Bit Test File, Skip if Set

Syntax: [label] BTFSF f,b,a

Operands: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$ Operation: skip if $(f < b) = 1$

Status Affected: None

Encoding:

1010	bbba	ffff	ffff
------	------	------	------

Description:

If bit 'b' in register 'r' is 1 then the next instruction is skipped.

If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'r'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFSF FLAG, 1, 0
 FALSE :
 TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;
 PC = address (FALSE)
 If FLAG<1> = 1;
 PC = address (TRUE)

BTG **Bit Toggle f**

Syntax: [label] BTG f,b,a

Operands: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$ Operation: $(f \ll b) \rightarrow f \ll b$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff
------	------	------	------

Description: Bit 'b' in data memory location 'f' is inverted. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: BTG PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [0x75]

After Instruction:

PORTC = 0110 0101 [0x65]

BV **Branch if Overflow**

Syntax: [label] BV n

Operands: $-128 \leq n \leq 127$ Operation: if overflow bit is '1'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn
------	------	------	------

Description: If the Overflow bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BV Jump

Before Instruction

PC = address (HERE)

After Instruction

If Overflow = 1;
PC = address (Jump)If Overflow = 0;
PC = address (HERE+2)

BZ Branch if Zero

Syntax: [label] BZ n
Operands: $-128 \leq n \leq 127$
Operation: if Zero bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0000	nnnn	nnnn
------	------	------	------

Description: If the Zero bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BZ Jump

Before Instruction

PC = address (HERE)

After Instruction

If Zero = 1;

PC = address (Jump)

If Zero = 0;

PC = address (HERE+2)

CALL Subroutine Call

Syntax: [label] CALL k,s
Operands: $0 \leq k \leq 1048575$
 $s \in [0,1]$

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(STATUS) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

1st word (k<7:0>)

2nd word (k<19:8>)

1110	110s	k ₇ k ₆ k ₅ k ₄	k ₃ k ₂ k ₁ k ₀
1111	k ₁₉ k ₁₈ k ₁₇ k ₁₆	k ₁₅ k ₁₄ k ₁₃ k ₁₂	k ₁₁ k ₁₀ k ₉ k ₈

Description:

Subroutine call of entire 2M byte memory range. First, return address (PC+4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>.	Push PC to stack	Read literal 'k'<19:8>. Write to PC
No operation	No operation	No operation	No operation

Example: HERE CALL THERE, Fast

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (THERE)

TOS = Address (HERE + 4)

WS = W

BSRS = BSR

STATUSS = STATUS

CLRF		Clear f					
Syntax:	[label] CLRF f,a						
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$						
Operation:	$000h \rightarrow f$ $1 \rightarrow Z$						
Status Affected:	Z						
Encoding:	<table border="1"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table>			0110	101a	ffff	ffff
0110	101a	ffff	ffff				
Description:	Clears the contents of the specified register. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).						
Words:	1						
Cycles:	1						
Q Cycle Activity:							
Q1	Q2	Q3	Q4				
Decode	Read register 'f'	Process Data	Write register 'f'				

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00

CLRWDT		Clear Watchdog Timer							
Syntax:	[label] CLRWDT								
Operands:	None								
Operation:	000h → WDT, 000h → WDT postscaler, 1 → \overline{TO} , 1 → \overline{PD}								
Status Affected:	\overline{TO} , \overline{PD}								
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr></table>					0000	0000	0000	0100
0000	0000	0000	0100						
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits \overline{TO} and \overline{PD} are set.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	Q1	Q2	Q3	Q4					
	Decode	No operation	Process Data	No operation					

Example: CLRWDT

Before Instruction
WDT counter = ?
After Instruction
WDT counter = 0x00
WDT Postscaler = 0
 \overline{TO} = 1
 \overline{PD} = 1

COMF **Complement f**

Syntax: **[label] COMF f,d,a**

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) \rightarrow \text{dest}$

Status Affected: N,Z

Encoding:

0001	11da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycl Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: COMF REG, 0, 0

Before Instruction

REG = 0x13

After Instruction

REG = 0x13

W = 0xEC

CPFSEQ **Compare f with W, skip if f = W**

Syntax: **[label] CPFSEQ f,a**

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(f) - (W)$,
skip if $(f) = (W)$
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff
------	------	------	------

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If $f = W$ then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSEQ REG, 0
NEQUAL :
EQUAL :

Before Instruction

PC Address = HERE

W = ?

REG = ?

After Instruction

If REG = W;

PC = Address (EQUAL)

If REG \neq W;

PC = Address (NEQUAL)

CPFSGT Compare f with W, skip if f > W

Syntax: [label] CPFSGT f,a

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(f) - (W)$,
 skip if $(f) > (W)$
 (unsigned comparison)

Status Affected: None

Encoding:

0110	010a	ffff	ffff
------	------	------	------

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
 Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSGT REG, 0
 NGREATER :
 GREATER :

Before Instruction

PC = Address (HERE)
 W = ?

After Instruction

If REG > W;
 PC = Address (GREATER)
 If REG ≤ W;
 PC = Address (NGREATER)

CPFSLT Compare f with W, skip if f < W

Syntax: [label] CPFSLT f,a

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(f) - (W)$,
 skip if $(f) < (W)$
 (unsigned comparison)

Status Affected: None

Encoding:

0110	000a	ffff	ffff
------	------	------	------

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).

Words: 1

Cycles: 1(2)
 Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSLT REG, 1
 NLESS :
 LESS :

Before Instruction

PC = Address (HERE)
 W = ?

After Instruction

If REG < W;
 PC = Address (LESS)
 If REG ≥ W;
 PC = Address (NLESS)

DAW Decimal Adjust W Register

Syntax: [label] DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> > 9] or [C = 1] then
(W<7:4>) + 6 → W<7:4>;
else
(W<7:4>) → W<7:4>;

Status Affected: C

Encoding:

0000	0000	0000	0111
------	------	------	------

Description: DAW adjusts the eight bit value in W resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example 1: DAW

Before Instruction

W = 0xA5
C = 0
DC = 0

After Instruction

W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction

W = 0xCE
C = 0
DC = 0

After Instruction

W = 0x34
C = 1
DC = 0

DECf Decrement f

Syntax: [label] DECf f,d,a

Operands: $0 \leq f \leq 255$
 $d \in \{0,1\}$
 $a \in \{0,1\}$

Operation: (f) - 1 → dest

Status Affected: C,DC,N,OV,Z

Encoding:

0000	01da	ffff	ffff
------	------	------	------

Description: Decrement register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: DECf CNT, 1, 0

Before Instruction

CNT = 0x01
Z = 0

After Instruction

CNT = 0x00
Z = 1

GOTO Unconditional Branch

Syntax: [label]. GOTO k

Operands: $0 \leq k \leq 1048575$

Operation: $k \rightarrow PC_{\langle 20:1 \rangle}$

Status Affected: None

Encoding:

1st word ($k_{\langle 7:0 \rangle}$)

2nd word ($k_{\langle 19:8 \rangle}$)

1110	1111	$k_7 k_6 k_5 k_4$	$k_3 k_2 k_1 k_0$
1111	$k_{19} k_{18} k_{17} k_{16}$	$k_{15} k_{14} k_{13} k_{12}$	$k_{11} k_{10} k_9 k_8$

Description: GOTO allows an unconditional branch anywhere within entire 2M byte memory range. The 20-bit value 'k' is loaded into $PC_{\langle 20:1 \rangle}$. GOTO is always a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' $\langle 7:0 \rangle$,	No operation	Read literal 'k' $\langle 19:8 \rangle$, Write to PC
No operation	No operation	No operation	No operation

HALT Halt Processor

Syntax: [label]. HALT

Operands: None

Operation: Processor halts execution after HALT instruction

Status Affected: None

Encoding:

Description:

0000	0000	0000	0001
------	------	------	------

While functioning in emulation mode, execution of the halt instruction will halt processor execution. Toggling the HALT pin or resetting ($MCLR = 0$) will bring the device out of halt. HALT instruction is not recognized in non-emulation modes.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	HALT

Example: GOTO THERE

After Instruction

PC = Address (THERE)

INCF Increment f

Syntax: [label] INCF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$

Status Affected: C,DC,N,OV,Z

Encoding:

0010	10da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: INCF CNT, 1, 0

Before Instruction

CNT = 0xFF
Z = 0
C = ?
DC = ?

After Instruction

CNT = 0x00
Z = 1
C = 1
DC = 1

INCFSZ Increment f, skip if 0

Syntax: [label] INCFSZ f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$,
skip if result = 0

Status Affected: None

Encoding:

0011	11da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f'. (default)

If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INCFSZ CNT, 1, 0
NZERO :
ZERO :

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT + 1
If CNT = 0:
PC = Address (ZERO)
If CNT \neq 0:
PC = Address (NZERO)

INFSNZ Increment f, skip if not 0

Syntax: [label] INFSNZ f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$,
skip if result $\neq 0$

Status Affected: None

Encoding:

0100	10da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default).
If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INFSNZ REG, 1, 0
ZERO
NZERO

Before Instruction

PC = Address (HERE)

After Instruction

REG = REG + 1

If REG \neq 0;

PC = Address (NZERO)

If REG = 0;

PC = Address (ZERO)

IORLW Inclusive OR literal with W

Syntax: [label] IORLW k.

Operands: $0 \leq k \leq 255$

Operation: $(W) .OR. k \rightarrow W$

Status Affected: N,Z

Encoding:

0000	1001	kkkk	kkkk
------	------	------	------

Description: The contents of W are OR'ed with the eight bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: IORLW 0x35

Before Instruction

W = 0x9A

After Instruction

W = 0xBF

IORWF Inclusive OR W with f

Syntax: [label] IORWF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$ Operation: (W) .OR. (f) \rightarrow dest

Status Affected: N,Z

Encoding:

0001	00da	ffff	ffff
------	------	------	------

Description: Inclusive OR W with register 'f'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: IORWF RESULT, 0, 1

Before Instruction

RESULT = 0x13
W = 0x91

After Instruction

RESULT = 0x13
W = 0x93**MOVF** M v f

Syntax: [label] MOVF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$ Operation: $f \rightarrow$ dest

Status Affected: N,Z

Encoding:

0101	00da	ffff	ffff
------	------	------	------

Description:

The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write W

Example: MOVF REG, 0, 0

Before Instruction

REG = 0x22
W = 0xFF

After Instruction

REG = 0x22
W = 0x22

MOVFF Move f to f

Syntax: `[label] MOVFF fs,fd`

Operands: $0 \leq f_s \leq 4095$
 $0 \leq f_d \leq 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)

2nd word (destin.)

1100	ffff	ffff	ffff _s
1111	ffff	ffff	ffff _d

Description:

The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination 'f_d' can also be anywhere from 000h to FFFh.

Either source or destination can be W (a useful special situation).

MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

Before Instruction

REG1 = 0x33
REG2 = 0x11

After Instruction

REG1 = 0x33
REG2 = 0x33

MOVLB Move literal to low nibble in BSR

Syntax: `[label] MOVLB k`

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow \text{BSR}$

Status Affected: None

Encoding:

0000	0001	kkkk	kkkk
------	------	------	------

Description:

The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example: MOVLB 5

Before Instruction

BSR register = 0x02

After Instruction

BSR register = 0x05

LFSR Move literal to FSR

Syntax: `[label], LFSR f,k`

Operands: $0 \leq f \leq 2$
 $0 \leq k \leq 4095$

Operation: $k \rightarrow \text{FSRf}$

Status Affected: None

Encoding:

1110	1110	00ff	k ₁₁ kkk
1111	0000	k ₇ kkk	kkkk

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: `LFSR 2, 0x3AB`

After Instruction

FSR2H = 0x03
 FSR2L = 0xAB

MOVLW Move literal to W

Syntax: `[label], MOVLW k`

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk
------	------	------	------

Description: The eight bit literal 'k' is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `MOVLW 0x5A`

After Instruction

W = 0x5A

MOVWF **M ve W to f**Syntax: **[label] MOVWF f,a**Operands: $0 \leq f \leq 255$
 $a \in [0,1]$ Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

0110	111a	ffff	ffff
------	------	------	------

Description: Move data from W to register 'f'. Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: **MOVWF REG, 0**

Before Instruction

W = 0x4F
REG = 0xFF

After Instruction

W = 0x4F
REG = 0x4F**MULLW** **Multiply Literal with W**Syntax: **[label] MULLW k**Operands: $0 \leq k \leq 255$ Operation: $(W) \times k \rightarrow \text{PRODH:PRODL}$

Status Affected: None

Encoding:

0000	1101	kkkk	kkkk
------	------	------	------

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.

None of the status flags are affected.

Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH:PRODL

Example: **MULLW 0xC4**

Before Instruction

W = 0xE2
PRODH = ?
PRODL = ?

After Instruction

W = 0xE2
PRODH = 0xAD
PRODL = 0x08

MULWF Multiply W with f

Syntax: `[label] MULWF f,a`

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(W) \times (f) \rightarrow \text{PRODH:PRODL}$

Status Affected: None

Encoding:

0000	001a	ffff	ffff
------	------	------	------

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH:PRODL

Example: `MULWF REG, 1`

Before Instruction

W = 0xC4
REG = 0xB5
PRODH = ?
PRODL = ?

After Instruction

W = 0xC4
REG = 0xB5
PRODH = 0x8A
PRODL = 0x94

NEGF Negate f

Syntax: `[label] NEGF f,a`

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow f$

Status Affected: N, OV, C, DC, Z

Encoding:

0110	110a	ffff	ffff
------	------	------	------

Description: Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.

Words: 1
Cycles: 1
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `NEGF REG, 1`

Before Instruction

REG = 0011 1010 [0x3A]

After Instruction

REG = 1100 0110 [0xC6]

NOP	N Operation			
Syntax:	[label] NOP			
Operands:	None			
Operation:	No operation			
Status Affected:	None			
Encoding:	0000 1111	0000 xxxx	0000 xxxx	0000 xxxx
Description:	No operation.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	No operation	No operation	No operation

Example:

None.

POP	P p T p of Return Stack			
Syntax:	[label] POP			
Operands:	None			
Operation:	(TOS) → bit bucket			
Status Affected:	None			
Encoding:	0000	0000	0000	0110
Description:	<p>The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.</p> <p>This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</p>			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	No operation	POP TOS value	No operation

Example:

POP
GOTO NEW

Before Instruction

TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction

TOS = 014332h
PC = NEW

PUSH **Push Top of Return Stack**

Syntax: [label] PUSH

Operands: None

Operation: (PC+2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101
------	------	------	------

Description: The PC+2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack.
This instruction allows to implement a software stack by modifying TOS, and then push it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC+2 onto return stack	No operation	No operation

Example: **PUSH**

Before Instruction

TOS	=	00345Ah
PC	=	000124h

After Instruction

PC	=	000126h
TOS	=	000126h
Stack (1 level down)	=	00345Ah

RCALL **Branch Subroutine**

Syntax: [label] RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1101	1nnn	nnnn	nnnn
------	------	------	------

Description: Subroutine call with a jump upto 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: **HERE RCALL Jump**

Before Instruction

PC	=	Address (HERE)
----	---	----------------

After Instruction

PC	=	Address (Jump)
TOS	=	Address (HERE+2)

RESET Reset

Syntax: `[label] RESET`

Operands: None

Operation: Reset all registers and flags that are affected by a MCLR reset.

Status Affected: All

Encoding:

0000	0000	1111	1111
------	------	------	------

Description: This instruction provides a way to execute a MCLR reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

Example: RESET

After Instruction

Registers = Reset Value
Flags* = Reset Value

RETFIE Return from Interrupt

Syntax: `[label] RETFIE s`

Operands: $s \in [0,1]$

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if $s = 1$
(WS) → W,
(STATUS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL, STATUS reg.

Encoding:

0000	0000	0001	000s
------	------	------	------

Description: Return from Interrupt. Stack is popped and Top of Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUS and BSRS are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1
Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example: RETFIE Fast

After Interrupt

PC = TOS
W = WS
BSR = BSRS
STATUS = STATUS
GIE/GIEH, PEIE/GIEL = 1

RLCF Rotate Left f through Carry

Syntax: [label] RLCF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

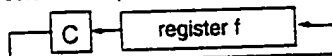
Operation: $(f \langle n \rangle) \rightarrow \text{dest} \langle n+1 \rangle$,
 $(f \langle 7 \rangle) \rightarrow C$,
 $(C) \rightarrow \text{dest} \langle 0 \rangle$

Status Affected: C,N,Z

Encoding:

0011	01da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLCF REG, 0, 0

Before Instruction

REG = 1110 0110
C = 0

After Instruction

REG = 1110 0110
W = 1100 1100
C = 1

RLNCF Rotate Left f (no carry)

Syntax: [label] RLNCF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f \langle n \rangle) \rightarrow \text{dest} \langle n+1 \rangle$,
 $(f \langle 7 \rangle) \rightarrow \text{dest} \langle 0 \rangle$

Status Affected: N,Z

Encoding:

0100	01da	ffff	ffff
------	------	------	------

Description:

The contents of register 'f' are rotated one bit to the left. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLNCF REG, 1, 0

Before Instruction

REG = 1010 1011

After Instruction

REG = 0101 0111

RRCF Rotate Right f through CarrySyntax: `[label] RRCF f,d,a`Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$ Operation: $(f \ll n) \rightarrow \text{dest} \ll n-1$,
 $(f \ll 0) \rightarrow C$,
 $(C) \rightarrow \text{dest} \ll 7$

Status Affect d: C,N,Z

Encoding:

0011	00da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RRCF REG, 0, 0`

Before Instruction

REG = 1110 0110
C = 0

After Instruction

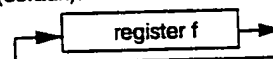
REG = 1110 0110
W = 0111 0011
C = 0**RRNCF Rotate Right f (no carry)**Syntax: `[label] RRNCF f,d,a`Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$ Operation: $(f \ll n) \rightarrow \text{dest} \ll n-1$,
 $(f \ll 0) \rightarrow \text{dest} \ll 7$

Status Affected: N,Z

Encoding:

0100	00da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: `RRNCF REG, 1, 0`

Before Instruction

REG = 1101 0111

After Instruction

REG = 1110 1011

Example 2: `RRNCF REG, 0, 0`

Before Instruction

W = ?
REG = 1101 0111

After Instruction

W = 1110 1011
REG = 1101 0111

SETF **Set f**

Syntax: `[label] SETF f,a`

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $FFh \rightarrow f$

Status Affected: None

Encoding:

0110	100a	ffff	ffff
------	------	------	------

Description: The contents of the specified register are set to FFh. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register †	Process Data	Write register †

Example: `SETF REG, 1`

Before Instruction

REG = 0x5A

After Instruction

REG = 0xFF

SLEEP **Enter SLEEP mode**

Syntax: `[label] SLEEP`

Operands: None

Operation: $00h \rightarrow WDT$,
 $0 \rightarrow WDT \text{ postscaler}$,
 $1 \rightarrow \overline{TO}$,
 $0 \rightarrow \overline{PD}$

Status Affected: \overline{TO} , \overline{PD}

Encoding:

0000	0000	0000	0011
------	------	------	------

Description: The power-down status bit (\overline{PD}) is cleared. The time-out status bit (\overline{TO}) is set. Watchdog Timer and its postscaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to sleep

Example: `SLEEP`

Before Instruction

$\overline{TO} = ?$
 $\overline{PD} = ?$

After Instruction

$\overline{TO} = 1^\dagger$
 $\overline{PD} = 0$

† If WDT causes wake-up, this bit is cleared

SUBFWB Subtract f from W with borrow

Syntax: [label] SUBFWB f,d,a

Operands: $0 \leq f \leq 255$

$d \in [0,1]$

$a \in [0,1]$

Operation: $(W) - (f) - (C) \rightarrow \text{dest}$

Status Affected: N,OV, C, DC, Z

Encoding:

0101	01da	ffff	ffff
------	------	------	------

Description: Subtract register 'f' and carry flag (borrow) from W (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

SUBFWB

Example 1: SUBFWB REG, 1, 0

Before Instruction

REG = 3

W = 2

C = 1

After Instruction

REG = FF

W = 2

C = 0

Z = 0

N = 1 ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction

REG = 2

W = 5

C = 1

After Instruction

REG = 2

W = 3

C = 1

Z = 0

N = 0 ; result is positive

Example 3: SUBFWB REG, 1, 0

Before Instruction

REG = 1

W = 2

C = 0

After Instruction

REG = 0

W = 2

C = 1

Z = 1

N = 0 ; result is zero

SUBLW Subtract W from literal

Syntax: `[label] SUBLW k`

Operands: $0 \leq k \leq 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

0000	1000	kkkkk	kkkk
------	------	-------	------

Description: W is subtracted from the eight bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example 1: `SUBLW 0x02`

Before Instruction

W = 1
C = ?

After Instruction

W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 0x02`

Before Instruction

W = 2
C = ?

After Instruction

W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 0x02`

Before Instruction

W = 3
C = ?

After Instruction

W = FF ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF Subtract W from f

Syntax: `[label] SUBWF f,d,a`

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - (W) \rightarrow \text{dest}$

Status Affected: N, OV, C, DC, Z

Encoding:

0101	11da	ffff	ffff
------	------	------	------

Description: Subtract W from register 'f' (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: SUBWF REG, 1, 0

Before Instruction

REG = 3
W = 2
C = ?

After Instruction

REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: SUBWF REG, 0, 0

Before Instruction

REG = 2
W = 2
C = ?

After Instruction

REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: SUBWF REG, 1, 0

Before Instruction

REG = 1
W = 2
C = ?

After Instruction

REG = FFh ; (2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

SUBWFB Subtract W from f with Borrow

Syntax: [label] SUBWFB f,d,a

Op rands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - (W) - (\bar{C}) \rightarrow \text{dest}$

Status Affected: N,OV, C, DC, Z

Encoding:

0101	10da	ffff	ffff
------	------	------	------

Description:

Subtract W and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

SUBWFB

Example 1: SUBWFB REG, 1, 0

Before Instruction

REG = 0x19 (0001 1001)
W = 0x0D (0000 1101)
C = 1

After Instruction

REG = 0x0C (0000 1011)
W = 0x0D (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

Example 2: SUBWFB REG, 0, 0

Before Instruction

REG = 0x1B (0001 1011)
W = 0x1A (0001 1010)
C = 0

After Instruction

REG = 0x1B (0001 1011)
W = 0x00
C = 1
Z = 1 ; result is zero
N = 0

Example 3: SUBWFB REG, 1, 0

Before Instruction

REG = 0x03 (0000 0011)
W = 0x0E (0000 1101)
C = 1

After Instruction

REG = 0xF5 (1111 0100) [2's comp]
W = 0x0E (0000 1101)
C = 0
Z = 0
N = 1 ; result is negative

SWAPF **Swap f**

Syntax: `[label] SWAPF f,d,a`

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f<3:0>) \rightarrow \text{dest}<7:4>$,
 $(f<7:4>) \rightarrow \text{dest}<3:0>$

Status Affected: None

Encoding:

0011	10da	ffff	ffff
------	------	------	------

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `SWAPF REG, 1, 0`

Before Instruction

REG = 0x53

After Instruction

REG = 0x35

TBLRD **Table Read**

Syntax: [label] TBLRD (*; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
 (Prog Mem (TBLPTR)) → TABLAT;
 TBLPTR - No Change;
 if TBLRD *+,
 (Prog Mem (TBLPTR)) → TABLAT;
 (TBLPTR) +1 → TBLPTR;
 if TBLRD *-,
 (Prog Mem (TBLPTR)) → TABLAT;
 (TBLPTR) -1 → TBLPTR;
 if TBLRD +*,
 (TBLPTR) +1 → TBLPTR;
 (Prog Mem (TBLPTR)) → TABLAT;

Status Affected: None

Encoding:

0000	0000	0000	10nn nn=0 *
			=1 *+
			=2 *-
			=3 +*

Description: There are four options with a TBLRD instruction to determine what happens to the 21-bit Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately, and the contents of the program memory location pointed to by the TBLPTR are loaded into the 8-bit Table Latch (TABLAT). The LSb of the TBLPTR selects which byte of the program memory location will be read. If LSb = 1, the high byte will be loaded into the TABLAT. If LSb = 0, the low byte will be loaded into the TABLAT.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (OE goes low) TABLAT updated

TBLRD **Table Read**

Example1: TBLRD *+ ;

Before Instruction

TABLAT	=	0x55
TBLPTR	=	0x00A356
MEMORY(0x00A356)	=	0x34

After Instruction

TABLAT	=	0x34
TBLPTR	=	0x00A357

Example2: TBLRD +- ;

Before Instruction

TABLAT	=	0xAA
TBLPTR	=	0x01A357
MEMORY(0x01A357)	=	0x12
MEMORY(0x01A358)	=	0x34

After Instruction

TABLAT	=	0x34
TBLPTR	=	0x01A358

TBLWT **Table Write**

Syntax: [label] TBLWT (*; *+; *-; +*)

Operands: None

Operation: if TBLWT *,
 (TABLAT) → Prog Mem(TBLPTR);
 TBLPTR - No Change;
 if TBLWT *+,
 (TABLAT) → Prog Mem(TBLPTR);
 (TBLPTR) +1 → TBLPTR;
 if TBLWT *-,
 (TABLAT) → Prog Mem(TBLPTR);
 (TBLPTR) -1 → TBLPTR;
 if TBLWT +*,
 (TBLPTR) +1 → TBLPTR;
 (TABLAT) → Prog Mem(TBLPTR);

Status Affected: None

Encoding:

0000	0000	0000	11nn
			nn=0 *
			=1 *+
			=2 *-
			=3 +*

Description:

There are four options with a TBLWT instruction. These options determine what happens to the Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately.

The contents of Table Latch (TABLAT) are written to the program memory location pointed to by TBLPTR.

If TBLPTR points to an external program memory location, then the instruction executes in two cycles.

Since the TABLAT is only one byte wide, a multiple of two TBLWT instructions must be executed to program internal memory locations. For example, if the device is defined to program one word at time, an internal memory location is programmed in the following manner:

- 1) Set TBLPTR to an even byte
- 2) Write low byte to TABLAT
- 3) Execute TBLWT *+ (2-cycle)
- 4) Write high byte to TABLAT
- 5) Execute TBLWT *+ (long write)

A long write to an internal EPROM location is terminated when an interrupt is received. The post increment TBLWT instruction is the only TBLWT instruction that is recommended for writes to internal memory. (Writes to internal EPROM are only available on devices with 64 or more pins.)

TBLWT **Table Write**

Cycles: 2 (many if long write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, WR goes low)

Example1:

TBLWT *+;

Before Instruction

TABLAT = 0x55
 TBLPTR = 0x00A356
 MEMORY(0x00A356) = 0xFF

After Instructions (table write completion)

TABLAT = 0x55
 TBLPTR = 0x00A357
 MEMORY(0x00A356) = 0x55

Example 2:

TBLWT *+;

Before Instruction

TABLAT = 0x34
 TBLPTR = 0x01389A
 MEMORY(0x01389A) = 0xFF
 MEMORY(0x01389B) = 0xFF

After Instruction (table write completion)

TABLAT = 0x34
 TBLPTR = 0x01389B
 MEMORY(0x01389A) = 0xFF
 MEMORY(0x01389B) = 0x34

Words:

TRAP **Debugger Subroutine Call**

Syntax: [label] TRAP

Operands: None

Operation: (PC) + 2 → TOS,
 000028h → PC<20:1>

Status Affected: INBUG

Encoding:

0000	0000	1110	0000
------	------	------	------

Description: Debugger Trap to 00028h. First, return address (PC+ 2) is pushed onto the return stack. Then the 20-bit value '000028h' is loaded into PC<20:1>. The INBUG status bit is set. TRAP is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decod	Push PC to stack	No operation	Write 000028h to PC
No operation	No operation	No operation	No operation

Example: HERE TRAP

Before Instruction

PC = Address (HERE)

After Instruction

PC = 000028h

TOS = Address (HERE + 2)

INBUG = 1

TRET **Trap Return from Subroutine**

Syntax: [label] TRET

Operands: None

Operation: (TOS) → PC
 PCLATU, PCLATH are unchanged

Status Affected: INBUG

Encoding:

0000	0000	1110	0001
------	------	------	------

Description: Return from debugger trap. The stack is popped and the top of the stack (TOS) is loaded into the program counter. The INBUG status bit is cleared.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack
No operation	No operation	No operation	No operation

Example: TRET

After Interrupt

PC = TOS

INBUG = 0

TSTFSZ Test f, skip if 0

Syntax: [label] TSTFSZ f,a

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: skip if f = 0

Status Affected: None

Encoding:

0110	011a	ffff	ffff
------	------	------	------

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
 Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE TSTFSZ CNT, 1
 NZERO :
 ZERO :

Before Instruction
 PC = Address(HERE)

After Instruction
 If CNT = 0x00,
 PC = Address (ZERO)
 If CNT ≠ 0x00,
 PC = Address (NZERO)

XORLW Exclusiv OR literal with W

Syntax: [label] XORLW k.

Operands: $0 \leq k \leq 255$

Operation: (W) .XOR. k → W

Status Affected: N,Z

Encoding:

0000	1010	kkkk	kkkk
------	------	------	------

Description: The contents of W are XOR'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: XORLW 0xAF

Before Instruction

W = 0xB5

After Instruction

W = 0x1A

XORWF Exclusive OR W with f

Syntax: [*label*]. XORWF f,d,a

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: (W) .XOR. (f) → dest

Status Affected: N,Z

Encoding:

0001	10da	ffff	ffff
------	------	------	------

Description: Exclusive OR the contents of W with register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in the register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: XORWF REG, 1, 0

Before Instruction

REG = 0xAF
W = 0xB5

After Instruction

REG = 0x1A
W = 0xB5